

AFRL-IF-RS-TR-2003-92
Final Technical Report
April 2003



PRINCIPLED ANALYSIS AND SYNTHESIS OF AGENT SYSTEMS USING TOOLS FROM STATISTICAL PHYSICS

Cornell University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K539

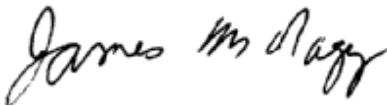
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.


The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-92 has been reviewed and is approved for publication.

APPROVED: 
JAMES M. NAGY
Project Engineer

FOR THE DIRECTOR: 
JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE APRIL 2003	3. REPORT TYPE AND DATES COVERED Final Jun 00 – Oct 02	
4. TITLE AND SUBTITLE PRINCIPLED ANALYSIS AND SYNTHESIS OF AGENT SYSTEMS USING TOOLS FROM STATISTICAL PHYSICS			5. FUNDING NUMBERS C - F30602-00-2-0596 PE - 62301E PR - TASK TA - 00 WU - 04	
6. AUTHOR(S) Bart Selman				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Cornell University 120 Day Hall Ithaca New York 14853-2801			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency AFRL/ITB 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2003-92	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: James M. Nagy/ITB/(315) 330-3173/ James.Nagy@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) The computational problem underlying the TASK (Taskable Agent Software Kit) domain is to find a solution to a large set of interacting, distributed constraints using a set of autonomous agents. The central hypothesis upon which our work is based is that large systems of distributed constraints behave much like physical systems with many interacting components. For example, sets of constraints can undergo phenomena such as sudden phase transitions from having many solutions to having no solution. Furthermore, analysis using the mathematical tools of statistical physics yields empirically-verifiable predictions that are stronger than those that can be obtained by a classical theoretical analysis. In this project, we (1) provided a complexity analysis for the agent RACE challenge problem, (2) developed a framework for fair bidding strategies in this domain, (3) demonstrated tradeoffs between aggressive and non-aggressive agents, (4) developed combinatorial auction test suites for agent system development, and (5) developed a series of software tools in support of large-scale agent software platforms.				
14. SUBJECT TERMS Intelligent Agent, Bidding Strategy, Combinatorial Auction, Taskable Agent Software Toolkit, TASK, Autonomous Agents			15. NUMBER OF PAGES 18	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

TABLE OF CONTENTS

1.	Summary	1
2.	Approach.....	1
3.	Problem Description (RACE Challenge Problem)	1
4.	Robustness and Sensitivity Analysis of Distributed Agent Architecture	4
5.	Structure in Market Mechanism Test Suite	7
6.	Software Developed.....	13
7.	References.....	13

TABLE OF FIGURES

Figure 1: Cost matrix for a sample problem.	2
Figure 2: Structure of the cost matrix for the uniform bidding case.....	2
Figure 3: Structure of the cost matrix for the uniform cost case.	3
Figure 4: Computational cost profile.	4
Figure 5: Computational cost profile.	6
Figure 6: Agressiveness level tradeoffs.	6
Figure 7. Median search cost for combinatorial auction when varying the number of cities for 2400 bids and different building penalties. Other parameters: goods: 12, max bid set: 5 and shipping cost: 1.8.	10
Figure 8. Median value for bid graph properties when varying the number of cities for and 2400 bids and different building penalties. Other parameters: goods: 12, max bid set: 5 and shipping cost: 1.8.	11
Figure 9. Median search cost when varying the shipping cost for 2400 bids. Other parameters: max bid set: 6.	11
Figure 10. Median value for bid graph properties when varying the shipping cost for 2400 bids. Other parameters: cities: 12, goods: 18, max bid set: 6, building penalty: 1.25.	12

1. Summary

The computational problem underlying the TASK (Taskable Agent Software Kit) domain is to find a solution to a large set of interacting, distributed constraints using a set of autonomous agents. The central hypothesis upon which our work is based is that large systems of distributed constraints behave much like physical systems with many interacting components. For example, sets of constraints can undergo phenomena such as sudden phase transitions from having many solutions to having no solution. Furthermore, analysis using the mathematical tools of statistical physics yields empirically-verifiable predictions that are stronger than those that can be obtained by a classical theoretical analysis.

In this project, we (1) *provided a complexity analysis for the agent RACE challenge problem*, (2) *developed a framework for fair bidding strategies in this domain*, (3) *demonstrated tradeoffs between aggressive and non-aggressive agents*, (4) *developed combinatorial auction test suites for agent system development*, and (5) *developed a series of software tools in support of large-scale agent software platforms*.

2. Approach

Our approach is based on our work on:

Connections between hard computational tasks and models from statistical physics[4, 6]. In this work we developed models based on statistical physics for hard combinatorial search problems. These models can be seen in terms of a distributed problem solving task where each individual constraint (or small group of constraints) is represented by a single agent. A natural mapping from the global behavior of searching for a solution to a physical system exists because in statistical physics global properties are determined by the microscopic interactions of the individual components. We extended this analysis to incorporate much richer agent models.

Randomization techniques. Randomization is a powerful mechanism to increase overall predictability and robustness. In classical combinatorial search problems, “rapid restarts” [3] and “portfolio” strategies [2] have been shown to be very effective. We extended these techniques to complex multi-agent systems to increase their overall robustness.

3. Problem Description (RACE Challenge Problem)

The RACE problem provides a compelling framework for the study of multi-agent behaviors and their inherent computational complexity. In this problem domain, the Department of Defense (DOD) needs to formulate and execute plans for transporting personnel and equipment in a case of an emergency situation. (For a current example, see <http://www.cnn.com/2003/US/02/09/sprj.irq.civil.fleet.action/index.html>.) A number of companies have signed agreements with the DOD to provide transportation required in

such situations. The agreements involve a certain amount of regular contractual work that the DOD awards to the companies with the understanding that in case of a crisis the companies provide emergency services proportional to the contractual work received. The companies provide estimates of what it would cost the company to provide each emergency job. After all estimates are in, the DOD goes through a job allocation process where it tries to assign jobs in a “fair” (balanced) manner to the various companies.

	j1	j2	j3	j4	j5
c1	100	100	50	50	50
c2	95	90	30	25	30
c3	95	90	25	30	25

Figure 1: Cost matrix for a sample problem.

We demonstrate the relation of the structure of the cost matrix (example in Figure 1) with the complexity of the allocation problem in two boundaries cases. In the "Uniform bidding" case, all companies declare the same cost for a given job. Figure 2 shows the structure of the cost matrix for this case, where colors represent different costs. The allocation problem for a decision version of min-max fairness becomes equivalent to bin packing. So, this case is NP-complete, although good approximation schemes and average-case results are known.

	J1	J2	J3
C3	Green	Blue	Red
C2	Green	Blue	Red
C1	Green	Blue	Red

Figure 2: Structure of the cost matrix for the uniform bidding case.

In the "Uniform cost" case, a company declares the same cost for all the jobs. Figure 3 shows the structure of the cost matrix for this scenario. Here we have a polynomial algorithm for the lex min-max allocation problem. So, this case represents a tractable sub-case of our general allocation problem.

	J1	J2	J3
C3			
C2			
C1			

Figure 3: Structure of the cost matrix for the uniform cost case.

We have also analyzed the typical (average) complexity of the problem when the cost matrix has a more general form, by studying the empirical computational complexity of solving the allocation problem when the cost matrix of the problem is obtained from random problem distributions, and we determined the existence of phase transitions in the decision version of the problem. A particular random problem distribution is defined by specifying the model for obtaining a cost matrix for the problem. For example, in one of the models, given a set of C companies and J jobs, we obtain the cost matrix by selecting, uniformly at random (u.a.r.), exactly C companies for every job such that the selected companies will be the only ones that can perform the job. The cost for every selected company is generated also u.a.r from a price interval $[LC, UC]$. We detected a phase transition in the decision version of min-max fairness allocation, i.e., is there an allocation such that the maximum cost paid by any company is less than a given upper limit cost K . The Phase transition occurs when we increase the ratio of the number of companies to the number of jobs. Figure 4 shows the results for an experiment for measuring the average complexity of the problem with different values for that ratio. The blue plot shows, for every different ratio companies/jobs, the average size of the search tree visited by a systematic search algorithm and the red plot the percentage of instances found to have solution with that ratio, from a total sample of 100 instances. We observe a phase transition from no solvable instances to solvable instances around a critical value of the ratio (0.56 in this experiment) and around this ratio we have also a peak in the computational complexity of solving the instances of the problem.

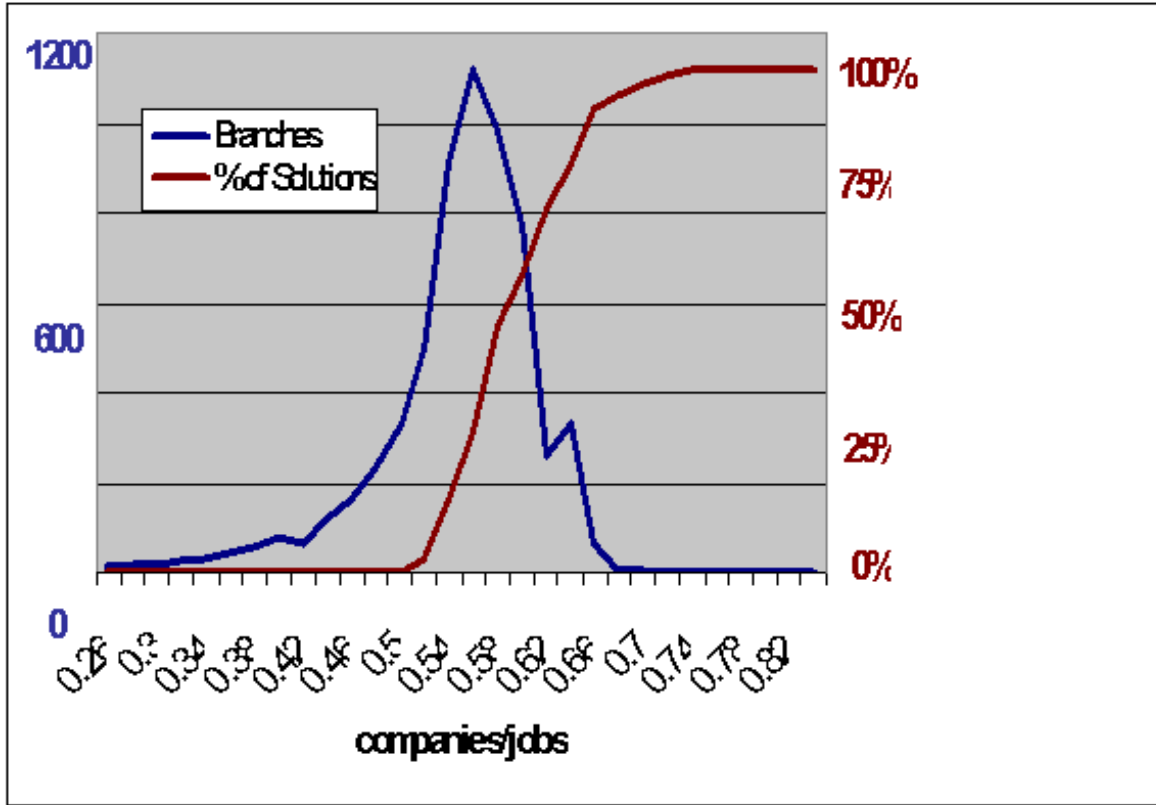


Figure 4: Computational cost profile.

We also analyzed the bidding process for companies submitting their costs for performing the various jobs and the connections of this problem with combinatorial auctions, because in a more general setting the companies will provide different costs for particular subsets of jobs, instead of giving costs for single jobs. We collaborated with the Stanford group working on the Combinatorial auction test suit CATS[5] to provide a understanding of the computational complexity of realistic combinatorial auction domains and applied this knowledge to get insight into the RACE domain.

4. Robustness and Sensitivity Analysis of Distributed Agent Architecture

In order to study the true dynamics of a practical agent system architecture, we analyzed a large series of experiments on the empirical agent server developed at the University of Michigan. The platform models an interactive market system in which agents compete for sets of goods via an auction mechanism.

The platform gives us a unique opportunity to experiment with general agent design methodologies. The scenario is very dynamic with multiple auctions running continuously. (This models many real world situations, where competition is an on-going activity.) In such dynamic situations, a very interesting tradeoff develops in which the agent has to repeatedly choose between waiting somewhat longer to acquire more information about price movement and competitors behavior versus acting quickly and acquiring goods and services as soon as possible. Waiting longer, increases the agent's

ability to determine the true value of goods based on its competitors behavior but the waiting itself leads to a gradual increase in prices and may even result in certain goods becoming unavailable.

In order to deal with the continuously changing dynamics of the agent interactions, we used a two-phase approach. In phase I, the agent computes what would be the *optimal bids to make given its current information about the prices and competitors behavior (the "plan")*. In phase II, the agent places a set of bids, based on the objectives computed in phase I. After phase II, the agent switches back to phase I and *re-computes its objectives, given any new information that has become available*. Phase I and II are tightly integrated and highly optimized. Each phase takes at most 1 to 3 seconds. We found that this rapid turnaround was absolutely paramount in developing an overall successful agent. Fast re-planning and bidding cycles appear to be the best way to deal with the dynamics of the interacting multi-agent domain.

The following graph (Figure 5) shows the performance increase ("Avg. score") of a series of agent bidding strategies (from non-aggressive to aggressive) agents in a pool of generic ("baseline") agents. We also studied the effect of "price modeling", as used in the planning phase (phase I). The price modeling approach incorporates a Bayesian scheme to predict price movements in the market. From the figure 5 we see that detailed price modeling and mildly aggressive to aggressive agent strategies lead to the best overall behavior.

Our next graph, Figure 6, shows how the various agent strategies perform when playing against a mix of agents. In particular, we studied each type of agent against a series of environments (ranging from less aggressive agents to more aggressive ones). It is encouraging to see that we obtained an *overall dominant* strategy with the medium aggressive agent. That is, that agent did well across the board, against a range of agents of different aggressiveness. Given that one has in general only limited information about ones opponents, and opponents can change strategies at any point in time, it is interesting that one can develop strategies that are *robust* in such different environments.

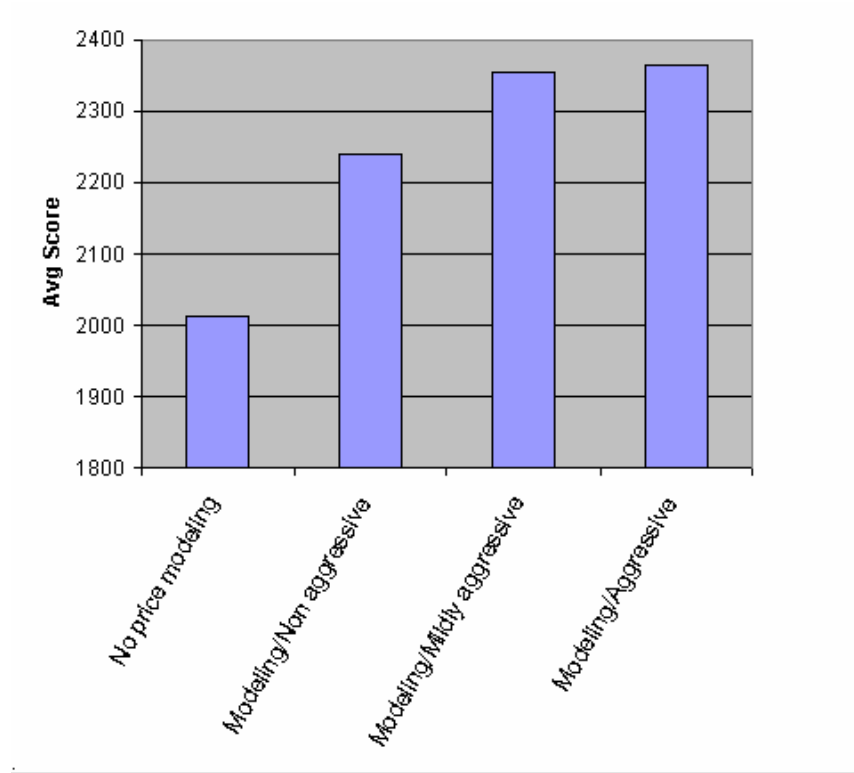


Figure 5: Computational cost profile.

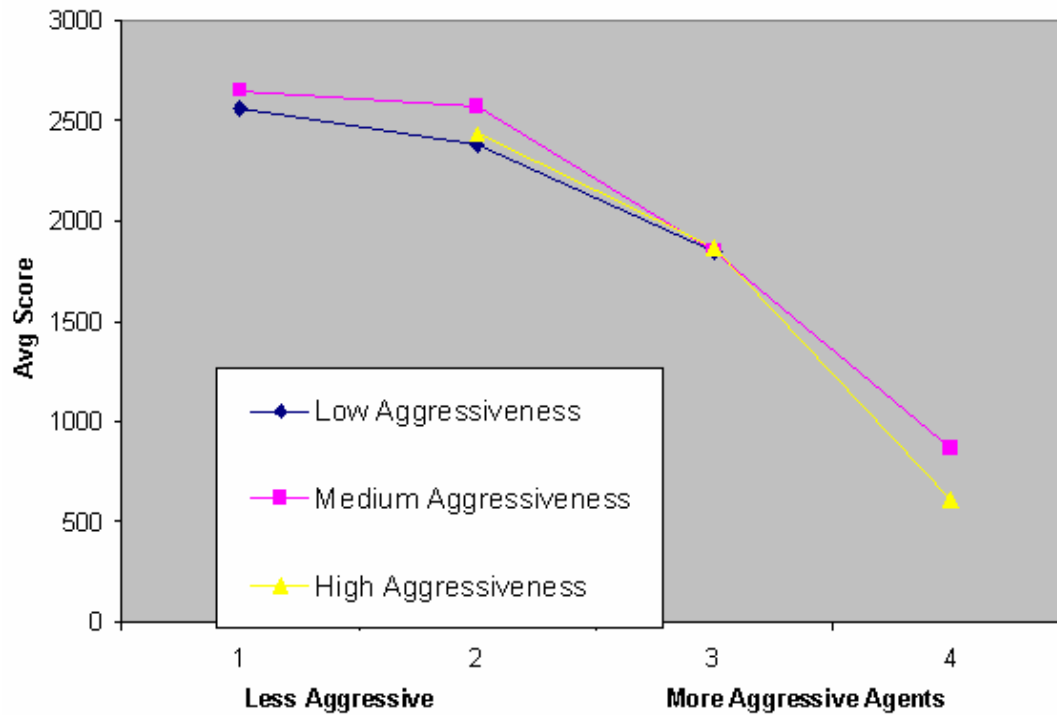


Figure 6: Aggressiveness level tradeoffs.

Our empirical agent toolkit enables the user to study a range of agent architectures, and the Michigan agent server allows for a detailed empirical comparison between

architectures. Of particular relevance are: *robustness, adaptability, and bidding and planning strategies.*

Our platform also enables groups to effectively compare their agent architectures. Interactions all take place through a uniform agent communication protocol and via a web server/client architecture, allowing for distributed and remote interactions. One very interesting issue that arises in this setup is that the delays in communications, although quite short in general, do become a key parameter of the agent architecture. In other words, the architectures and agent mechanisms need to have effective ways of dealing with unexpected delays. Communication delays are of course a key issue in distributed systems in general. Agent-based interaction protocols provide a new framework of studying and handling such delays robustly.

5. Structure in Market Mechanism Test Suite

We conducted an extensive analysis of the combinatorial agents auction test suite to obtain a detailed picture of the structure and complexity tradeoffs. This part of our overall effort was based on collaboration with the Stanford group, as suggested by Dr. Hendler. Regular meetings (physical and by phone) took place between the groups.

Combinatorial auctions (CA) are becoming an important mechanism for effective negotiations between autonomous distributed agents. For that reason, algorithms for winner determination are needed in order to manage markets of considerable size. At the same time, we need to understand what makes a CA instance more difficult to solve for a CA algorithm. At the present time, realistic CA instance distributions --- capturing realistic multi-agent interactions --- have been defined, but we do not have a clear way of generating instances with different computational properties, as is needed for the development of better agent interaction protocols.

In the context of Constraint Satisfaction Problems (CSPs), it has been shown that the typical complexity of solving them is correlated with certain global properties of the underlying constraint graph of the problem. We have uncovered that there is also a correlation between the computational hardness of a CA instance and certain properties of the associated *bid graph*. The bid graph of a CA instance is a graph where the vertices represent dominant bids and the edges link the vertices associated with bids that share some good. For showing this relation between (typical) computational hardness and bid graph properties, we have selected a CA instance distribution taken from the Combinatorial Auction Test Suite (CATS, test suite developed at Stanford). This suite is the first, and as far as we know unique, collection of CA instance distributions defined from realistic market problems. For generating CA instances from the different market problems, every instance generator of CATS has a different list of parameters that we can tune in order to generate instances that reflect particular market conditions. However, as we will see, not all combinations of parameter values produce instances with the same hardness. So, if we are interested in recognizing the setting that produces the hardest instances with a particular CA instance distribution, we should use some method for

tuning the hardness. We propose the study of the correlation between hardness and bid graph properties, as a first step in that direction.

Our main results indicate that the number of vertices and the edge density of the bid graph have a clear correlation with the computational hardness of a CA instance. A third property, the clustering coefficient, also shows some correlation, although not so strong as the two others.

The bid graph

The bid graph of a combinatorial auction instance, is defined as a graph (V, E) where:

The set V is the set of dominant bids. We say that a bid A dominates a bid B if the set of goods in A is a subset (proper or not) of the set of goods in B and the bid price of A is higher than the price of B. A bid that does not dominate any bid and that is not dominated by any other bid is also considered as a dominant bid. The bids that are dominated by other bids are not considered in the bid graph. (Note that such bids would never be selected anyway.)

The set E contains an edge between two vertices if the corresponding bids have a common good. So, a winning (maximizing) valid selection of dominant bids represents an independent set of the bid graph.

There are different properties of the bid graph that intuitively can be seen to have an impact in the search for the optimal solution. These are the ones we have investigated:

Number of vertices. Because the vertices represent dominant bids, usually this number will be much smaller than the original number of bids submitted in the CA instance. This quantity is important because represents the real number of bids that the search algorithm will have to consider.

Edge density. Is the ratio of the number of edges of the bid graph to the number of edges of a complete graph with the same number of vertices as the bid graph.

Clustering coefficient. Is as measure of the cliqueness between local neighbors in the graph. For a vertex with k neighbors, then at most $k(k-1)/2$ edges can exist between them (this occurs if they form a k -clique). The clustering of a node is the fraction of the allowable edges that occur. The clustering coefficient is the average clustering over all the vertices of the graph. Observe that a set of k XOR bids will form a k -clique, but other cliques can be formed because of the relations between bids submitted by different bidders.

The paths problem

The "Paths" problem models a particular bidding scenario that arises when bidders try to purchase "connections" between points of interest. For example, in a communication

domain we have a set of locations of interest that need to be connected via network owned or controlled by a single company or conglomerate. Between some of the locations, there exists a (geographical) connection that allows the company to install a fiber link between them. If a company wants to connect two remote locations, it should find a path of intermediate connected locations between the two desired locations. So, in this problem, companies bid for particular paths in a locations graph, where edges represent locations "directly" connected.

For generating an instance of the Paths problem, the generator first creates the locations graph. Every location is a city. Once the locations graph is created, a set of bids is generated.

Results for Combinatorial Auctions

We can divide the set of parameters of the Paths problem in two different sets:

Locations graph parameters. Number of building paths, initial connections, building penalty, number of cities and number of edges (goods).

Bid graph parameters. Number of total bids and shipping cost factor.

In the experiments describe here, we have used the default value for the number of building paths provided by Paths (namely, $\text{cities}^{2/4}$). This number of building paths was sufficient to construct the full location graph. Although the generator has the option to create the locations graph through an annealing process using the building penalty, we use a fixed building penalty to investigate its impact on the complexity of solving the resulting CA instances. In all the experiments, the number of instances per data point is 25.

To modify the structure of the locations graph, we have considered changing the ratio of the number of cities to goods and the building penalty.

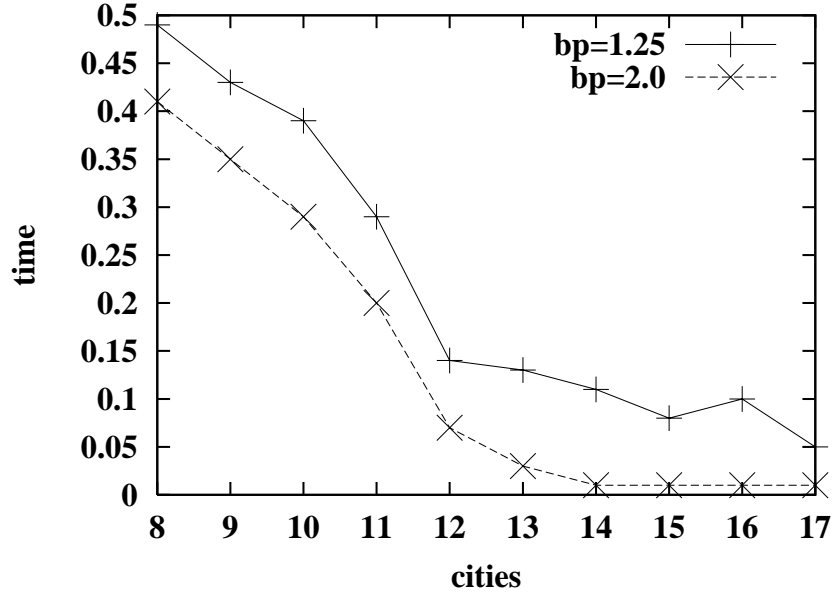


Figure 7. Median search cost for combinatorial auction when varying the number of cities for 2400 bids and different building penalties. Other parameters: goods: 12, max bid set: 5 and shipping cost: 1.8.

Figure 7 shows the results for CA instances where we have fixed all the parameters except for the number of cities. The graph shows the median time complexity when solving instances with a particular value of the building penalty (1.25 or 2.00) while modifying the number of cities. We observe clearly that a lower ratio of goods to cities produces computationally easier instances, and that given the same ratio, a smaller building penalty increases uniformly the hardness of the instance. (Note that the number of goods is fixed. So, a larger number of cities decreases the "goods per city ratio".)

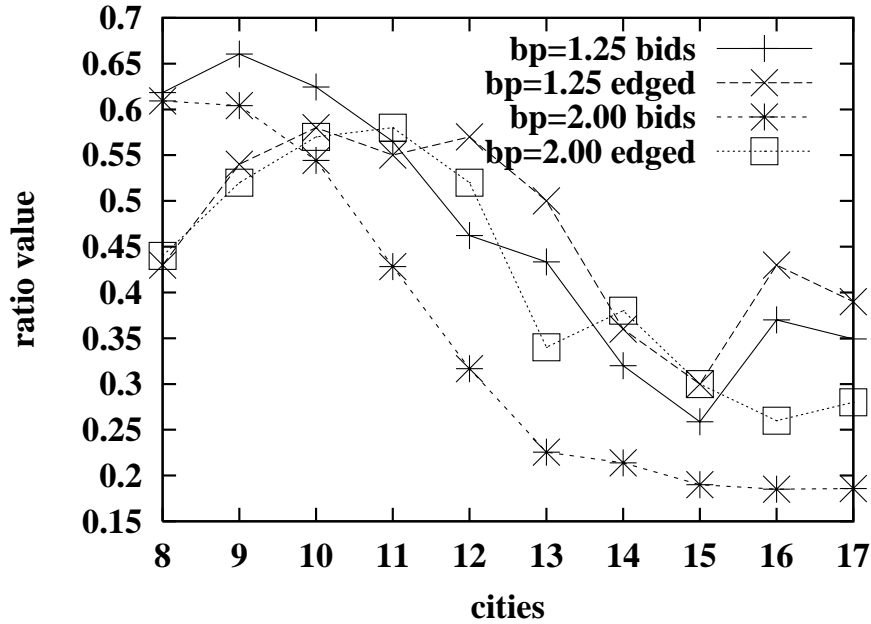


Figure 8. Median value for bid graph properties when varying the number of cities for and 2400 bids and different building penalties. Other parameters: goods: 12, max bid set: 5 and shipping cost: 1.8.

We then considered the relation between the median hardness and the properties of the CA instances by looking at some of the *structural* properties of their bid graphs. Figure 8 shows the median value of the number of vertices of the bid graph (normalized by the total number of submitted bids) and the edge density. We observe a correlation between the complexity and the number of vertices of the bid graph. For the edge density we observe that the easiest instances coincide with the instances with lower edge density. More experiments with a higher ratio of edges to cities should be performed in order to locate more precisely the hardest instances and their bid graph properties.

Modifying the structure of the bid graph

We also need to consider the effect of modifying those parameters that will change the bid generation process once we have generated the locations graph, i.e., those parameters that will have a direct effect on the resulting bid graph without modifying the locations graph.

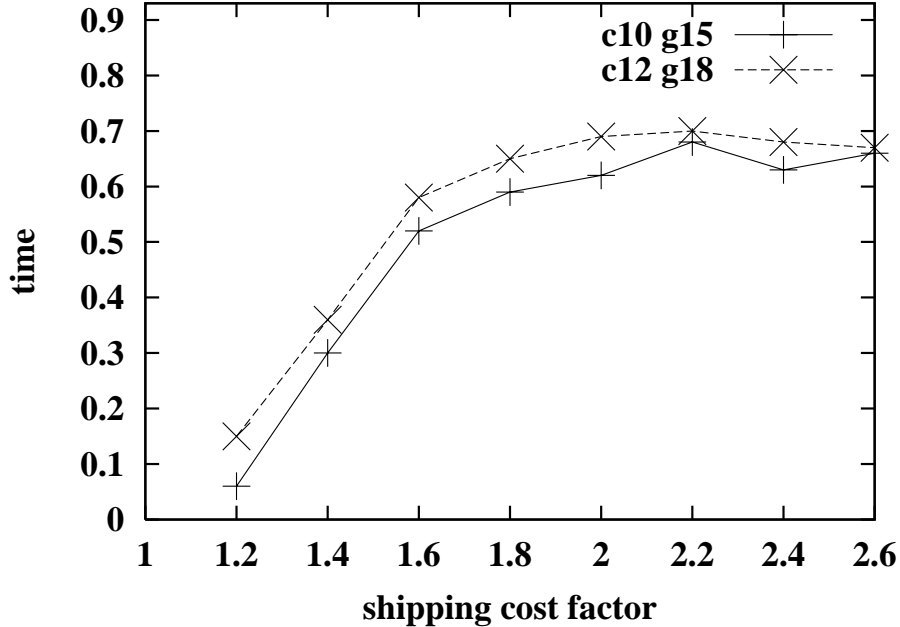


Figure 9. Median search cost when varying the shipping cost for 2400 bids. Other parameters: max bid set: 6.

Figure 9 shows results for CA instances where we have fixed all the parameters except for the shipping cost factor. This factor has a direct effect in the bid graph, because the larger this parameter, the larger the bids that will be submitted by the bidders, although the max bid set parameter sets up an upper limit in terms of the number of possible different bids a bidder can submit. Therefore, even with a large increase in the shipping

cost factor, the max bid set parameter bounds the number of paths with different prices a bidder can submit as a set of XOR bids; the lowest ones always being the first submitted.

The combinations of number of cities and number of goods chosen, are such that the edge density for the locations graph is 0.33 with 10 cities and 15 goods, and with 12 cities and 15 good the edge density is 0.27. By looking at the results, we observe that a computational cost increase, when we increase the shipping cost factor.

We also observe in these experiments a correlation between complexity and bid graph properties. Figure 4 shows the median value of the normalized number of vertices of the bid graph, the edge density, and the clustering coefficient. We observe a correlation between the complexity and the number of vertices of the bid graph, in such a way that the number of vertices increases until the point where we have the sharp increase in the complexity, but from then on decreases somewhat. For the clustering coefficient, we observe an inverse correlation, *i.e.*, the greater value for the clustering coefficient is found for the easiest instances. This is reasonable because the more clustered the bids are, the more pruning can be performed by a branch and bound search method.

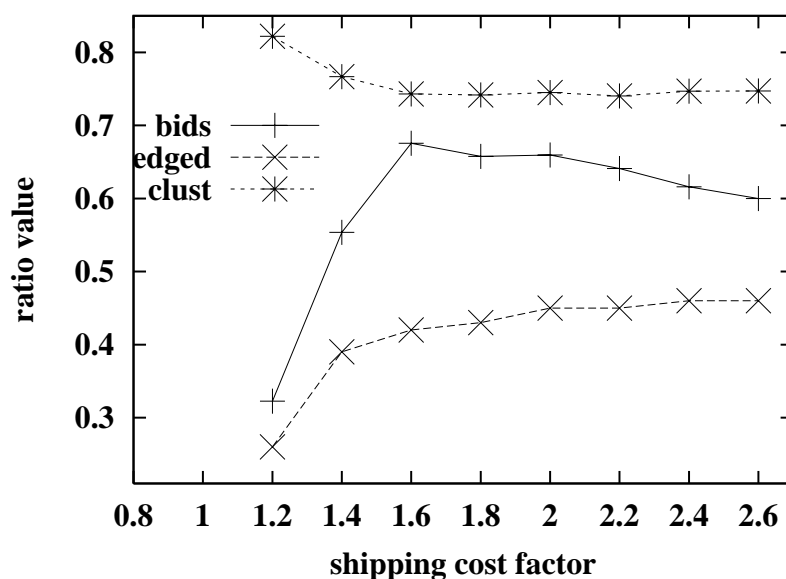


Figure 10. Median value for bid graph properties when varying the shipping cost for 2400 bids. Other parameters: cities: 12, goods: 18, max bid set: 6, building penalty: 1.25.

Based on our empirical study of the combinatorial auction agent interaction mechanisms, we have identified a number of parameters that capture the computational properties of the overall market mechanism.

A key parameter is the *edge density* in the location graph: easier instances occur at lower densities. This is consistent with our intuition about under-constrained instances in classical search problems. Our results also show that the building penalty (penalty for "construction of new goods") is a factor that uniformly changes the complexity of the CA instance obtained.

6. Software Developed

A series of software tools were developed as part of this effort:

Agent Toolkit with plug-and-play bidding agents. The toolkit is enabled to interact over the internet, allowing other groups to easily interact with and use the platform.

Structure and scaling analysis tools. These tools allow the user to study underlying structure and computational complexity scaling issues in agent platforms.

Combinatorial auction bid allocation tools. Tools are based on linear programming methods (CPLEX) combined with combinatorial search strategies.

7. References

1 R. Bejar, I. Vetsikas, C. Gomes, Henry Kautz and B. Selman. **Structure and Phase Transition Phenomena in the VTC Problem.**

In *TASK PI Meeting Workshop*, 2001.

2 Carla P. Gomes and Bart Selman. **Algorithm portfolio design: Theory vs. practice.**

In *Proc. of UAI-97*, 1997.

3 Carla P. Gomes, Bart Selman, and Henry Kautz. **Boosting combinatorial search through randomization.**

In *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI'98, Madison/WI, USA*, pages 431-437. AAAI Press, 1998.

4 R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. **Determining computational complexity from characteristic 'phase transitions'.**

Nature, 400(8), 1999.

5 Leyton-Brown, K., Pearson, M., & Shoham, Y. **Towards a Universal Test Suite for Combinatorial Auction Algorithms**

In the Proceedings of ACM Conference on Electronic Commerce (EC-00), 2000.

6 Bart Selman and Scott Kirkpatrick. **Critical behavior in the computational cost of satisfiability testing.**

Artificial Intelligence, 81(1-2):273-295, 1996.